

Protecting Sensitive Data

with MySQL and PHP

Conférence PHP Québec

March 20, 2003. Montréal

Zak Greant <zak@mysql.com>

Setup

- o [Dazzling Introduction]
- o [Cue Laugh Track at Intervals]
- o [Insert Footage of Screaming Crowd]
- o [Proceed Calmly to Next Slide :)]

Feeling Nervous Now?

- o Good!
- o Sensitive data is never safe!

The CSI 2002 Computer Crime and Security Survey

- o Surveyed ~500 large companies (generally million to billion dollar income range)
- o 90% reported security breaches
- o 80% acknowledged financial losses
- o Most frequent point of attack: 74% Internet / 33% Intranet
- o 34% reported the intrusion to law enforcement
- o 12% reported theft of transaction information

Credits

- o Peter Wayner
- o Translucent Databases
- o A. Menezes, P. van Oorschot, S. Vanstone
- o CRC Press
- o Handbook of Applied Cryptography
- o PHP Development Team

Thank You

- o Conference Organizers and Sponsors
- o Damien Seguy
- o Calgary PHP User Group
- o Derick Rethans
- o ... and thanks for attending!

Overview/Topics

- o One hour rapid overview
- o What sensitive data is
- o The risks associated with managing sensitive data
- o Common methods to protect sensitive data
- o Methods to protect sensitive data that integrate the protections with the data itself
- o Basic tools for implementing the embedded protections
- o Usage notes for each of the tools
- o Sample code

What is Sensitive Data?

- o A broad definition is: "Data that requires constraints on how it is consumed."
- o - with mustard ("Do you have any Grey Poupon?")
- o - by whom
- o - when
- o - how
- o This is probably too general!
- o Here are some major sensitive data issues

Identity Crisis

- o Personal data is used daily to facilitate many activities.
- o Monetary exchanges
- o Junk mail delivery
- o Background and credit checks

Credentials

- o Credentials derived from personal data are often used as the primary proof of identity.
- o Identity information is used to automate consent.
- o Theft of identity information becomes theft of identity and consent.
- o This ties to many, many issues. Competitive advantage, safety, privacy, ...

Reputation Poisoning

- o Someone pretends to be you
- o Financial losses (press release fraud)
- o Personal danger (cyber stalking)
- o Damage to implicit credentials
- o ...or someone abuses embarrassing or compromising information about you
- o Medical records
- o Extra-curricular activities
- o Schedules/locations

You Get The Idea

- o Disclosure of sensitive information is bad...
- o don't disclose sensitive data.

Risks/Costs of Managing Secure Data

- o Legal liability (responsibility for data, governing bodies, health boards, etc.)
- o Maintenance hassle (storage, policies, audits)
- o Sensitive data is attractive to thieves
- o Sensitive data storage compromises the freedom of others

Broad Vulnerabilities

- o Human - trusted user abuse, lack of vigilance, uneducated users (joe, PHB syndrome)
- o Network - virtual/physical compromise
- o Server - virtual/physical access
- o Software - permissions escalation, software interactions, side effects
- o Storage - disk, hardcopy
- o Consumption - data consumer disclosure, transfer to insecure area, poor data management...

Solutions

- o Combine methods!
- o Restrict access
- o Encrypt communications
- o Maintenance (audits, security patches, training, incremental application improvements)
- o Implement data management/consumption protocols
- o Educate data consumers
- o Encrypt the data
- o Render data unreadable (but still useable)
- o Discard data when not needed
- o Distribute information

Specific Focus

- o This session focuses on protecting the data with cryptographic methods rather than access controls
- o Restricting application and data access is critical, but...
- o many developers and software tools already deal with access control issues fairly well (Apache, Linux, MySQL, PHP, ...)
- o Data is vulnerable at so many points that it takes massive effort to create a very low risk environment for it if only relying on access control and policies

Drawbacks

- o Not everything is wonderful about these techniques
- o The data is more difficult to manipulate and process
- o Data can be lost forever
- o Users may find that the application is less convenient to use
- o Additional resources may be required to support the application
- o The application may be harder to debug

Advantages

- o Data is much less vulnerable even if leaked
- o Reduced likelihood of problems with failures in policy
- o Moderate combinations of AC restrictions and data encryption/manipulation require much less effort than extensive access control schemes and offer much greater security
- o Reduced risk

Tools

- o Quantization
- o Data Poisoning / Decoys
- o Crypto Hashes
- o Public/Private Key Encryption (RSA, DSA / AES Rijndael, Blowfish, 3DES)
- o Public/Private Key Crypto Schemes (Diffie/Hellman / Shared Key Segments)

Quantization

- o Mapping a data set onto a smaller data set
- o Reducing the amount of sensitive data disclosed
- o ...while still retaining useful information.

Examples

- o Lossy compression or scaling of images or signals to allow for broader distribution
- o Discarding postal code or IP address information to prevent easy identification of individuals
- o Aggregating feedback from all school districts to reduce politicking

Data Poisoning, Bait and Decoys

- o Poisoned data makes compromise of the real data more expensive.
- o Bait is false data whose usage can be tracked, allowing administrators to detect and "watch" abusers.
- o Decoy data disguises real information

Examples

- o Artificial user data that includes genuine contact information that is monitored. (good for employee and paper leaks)
- o Artificial user data that triggers warning systems if accessed.
- o Disguise the data entry requested by a user by mixing it with non-requested data.
- o Generate additional entries in systems where the cost of compromise is relatively low. (Same data, modify keys)

Crypto Hashes

- o Algorithms that output a unique 'fingerprint' for a given input of data (MDx, SHA1)
- o Input data may be of almost any length
- o No key is required - only data is needed as input (Though an IV can be used in some cases (! PHP))
- o Generated "fingerprint" is of a fixed length
- o Deterministic
- o Considered nigh-impossible to reconstruct the input data given the output data
- o Also called one-way functions, message digests, signatures and hashes

Examples

- o Unix, MySQL, htpasswd, ... password systems
- o Hash user credentials together with preference information
- o i.e. md5(\$user_name, \$password, \$mark_on_test)
- o Data is not reversible - just comparable

Public Key Encryption

- o RSA/DSA are the most common asymmetric ciphers (used in SSH and OpenSSL)
- o Uses a pair of keys for encryption and decryption
- o Encryption is accomplished with a public key
- o Decryption can only be done with a private key
- o Much slower than crypto hashes
- o Still slower than most private (symmetric) key ciphers
- o Tremendously useful for protecting data and
- o Relatively easy to use

Examples

- o Encrypting a message digest of a clear-text document
- o Securing data entered into a database on an insecure server

Private Key Encryption

- o AES, 3DES and Blowfish are some of the most common symmetric ciphers
- o Uses a single private key for encryption and decryption
- o Slower than crypto hashes
- o Most private key ciphers are much faster than most public key ciphers
- o Also tremendously useful for protecting data and
- o Relatively easy to use

Examples

- o Examples in the code up ahead

Key Exchange Schemes

- o Allow keys to be shared in ways that reduce the possibility of compromise.
- o Public/Private key exchanges that allow users to exchange public keys that are used to generate a shared private key (Diffie/Hellman)
- o Distributed private key exchanges (see code)

MySQL

- o SHA1
- o MD5
- o PASSWORD (caveat)
- o ENCRYPT (UNIX crypt)
- o ENCODE/DECODE (trivial secrets only)
- o AES_ENCRYPT/DECRYPT (coming soon)
- o DES_ENCRYPT/DECRYPT (coming soon)
- o SSL Support (coming soon)
- o UDFs
- o Watch out for logging!

PHP

- o MCRYPT (Access to private/symmetric ciphers)
- o MHASH (Access to crypto hash algorithms)
- o OpenSSL (public/asymmetric ciphers (RSA/DSA))
- o BCMath and GMP (Large integer math)
- o Crack (Brute force password challenges)
- o General (String (md5, sha1), Regex, Array, Pack/Unpack, ...)

Example Code

- o Ensure that all users are made aware of a message with roughly equal opportunity
- o User enters a message intended for n people to read
- o User enters a list of recipients and a message
- o Application generates passphrase and a list of shared keys
- o Message is encrypted with the passphrase
- o Keys are distributed to users
- o Passphrase and keys are destroyed

Code

```

<?php

include 'MyConnect.php';
include 'Cipher.php';
include 'SharedKeys.php';

ob_start();

mysql_query( "DROP TABLE IF EXISTS equal_opps" );

$query = <<<EOS
CREATE TABLE equal_opps (
    id          INT UNSIGNED NOT NULL AUTO_INCREMENT,
    sender      VARCHAR(64) NOT NULL,
    recipients  TEXT NOT NULL,
    message     BLOB NOT NULL,
    PRIMARY KEY (id)
)
EOS;

mysql_query($query)
    or die("Could not execute query '$query' : " . mysql_error());

$message = <<<EOS
Sam Poblano from NetTech books is looking for an
author for a book titled 'MySQL and PHP for
Southern Germans'.
The job details are ...
EOS;

$key = "Usually Random!";

$cipher = new Cipher($key);

$recipients = array(
    'georg@php.net',
    'hartmut@php.net',
    'sandro@zoss.com'
);

$query = sprintf(
    "INSERT equal_opps (sender, recipients, message) VALUES ('s', 's', '%s')",
    'zak@mysql.com',
    join( ',', $recipients ),
    mysql_escape_string( $cipher->Encrypt($message) )
);

mysql_query( $query )

```

```

    or die("Could not execute query '$query'");
$query = "SELECT * FROM equal_opps";

echo $query, "\n", str_repeat('-', strlen($query)), "\n";
$qh = mysql_query( $query )
    or die("Could not execute query '$query'");

foreach( mysql_fetch_assoc( $qh ) as $k => $v ){
    printf( "-12s: s\n", $k, $v );
}

echo "\n\n", $msg = "Keys Generated for Recipients", "\n",
    str_repeat('-', strlen($msg)), "\n";
$keys = SharedKeys::Create( $key, count($recipients) );
foreach($keys as $index => $key){
    printf( "-18s: s\n", $recipients[$index], $key );
}

$msg = "Re-assembling Keys using 'SharedKeys::Assemble( array(\"".
    join("\n\n", $keys) ."\") );'";

echo "\n\n", $msg, "\n";
echo "Reassembled Key is: ", SharedKeys::Assemble( $keys );

$qh = mysql_query( "DROP TABLE equal_opps" );
$cipher->Cleanup();
$out = ob_get_contents();
ob_end_clean();
echo nl2br($out);
?>

```

More Example Code

```

<?php
class SharedKeys{
    function Create($secret, $people, $show = false){
        $step = 1;

        # Truncate/Pad $secret to 16 characters
        $secret = sprintf( "%-' 16s", substr($secret, 0, 16) );
        $keys = array();

        # Generate an array of random keys
        if( $show ){
            echo "\nGenerating Random Keys\n";
        }
        for( $n = 1; $n < $people; ++$n ){
            # Create the binary string from an md5 hash
            # Use mt_rand() to seed the hash
            $keys[] = pack("H*", md5(mt_rand()));

            if( $show ){
                echo "Generating Random Key: ", $keys[count($keys) - 1],"\n";
            }
        }

        # Generate the final key by XORing the
        # previous keys together with the secret
        if( $show ){
            echo "\nDeconstructing Original Key\n",
                "Deconstructing Original Key (Step 1): ", $secret, "\n";
        }

        $out = $secret;
        foreach( $keys as $key ){
            $out ^= $key;
            if( $show ){
                echo "Deconstructing Original Key (Step ", ++$step, "): " , $out, "\n";
            }
        }

        array_push($keys, $out);

        # Don't always allow $out to come last in the array
        sort($keys);
        return $keys;
    }

    function Assemble($keys, $show = false){
        $step = 0;
        $out = array_pop($keys);
        if( $show ){
            echo "\nReconstructing Original Key\n",
                "Reconstructing Original Key (Step ", ++$step, "): " , $out, "\n";
        }

        foreach($keys as $key){
            $out ^= $key;
            if( $show ){
                echo "Reconstructing Original Key (Step ", ++$step, "): " , $out, "\n";
            }
        }
        return $out;
    }
}
?>

```

Output

Yet More Example Code

```

<?php

class Cipher{
    var $cipher,
        $data,
        $iv,
        $key,
        $mode,
        $td;

    function Cipher( $key, $cipher=MCRYPT_RIJNDAEL_256, $mode=MCRYPT_MODE_CFB,
    $cipher_dir='',
        $mode_dir='' ){
        $this->key = $key;
        $this->cipher = $cipher;
        $this->mode = $mode;
        $this->cipher_dir = $cipher_dir;
        $this->mode_dir = $mode_dir;

        # Open the modules to support the chosen cipher and mode
        $this->td = mcrypt_module_open( $this->cipher, $this->cipher_dir,
    $this->mode,
        $this->mode_dir );

        # Create an initialization vector for the open modules from a random source
        $this->iv = mcrypt_create_iv( mcrypt_enc_get_iv_size($this->td),
    MCRYPT_DEV_RANDOM );
    }

    function init(){
        # Initialize the buffers required for encryption/decryption
        mcrypt_generic_init ( $this->td, $this->key, $this->iv);
    }

    function Encrypt( $data ){
        $this->init();
        return mcrypt_generic( $this->td, $data );
    }

    function Decrypt( $encrypted_data ){
        $this->init();
        return mdecrypt_generic( $this->td, $encrypted_data );
    }

    function Cleanup(){
        mcrypt_generic_deinit( $this->td );
        mcrypt_module_close( $this->td );
    }
}

/* Cipher Test
$cipher = new Cipher('boo');
$enc_data = $cipher->Encrypt("J. Random Data");
echo "Encrypted data: ", $enc_data, "\n";

$dec_data = $cipher->Decrypt($enc_data);
echo "Decrypted data: ", $dec_data, "\n";

$cipher->Cleanup();
*/
?>

```

Questions?

- o ...

Index

Setup	2
Feeling Nervous?	3
Credits	4
Thank You	5
Overview	6
What is Sensitive Data?	7
Identity Crisis	8
Reputation Poisoning	9
You Get The Idea	10
Risks/Costs	11
Broad Vulnerabilities	12
Solutions	13
Specific Focus	14
Drawbacks	15
Advantages	16
Tools	17
Quantization	18
Data Poisoning, Bait and Decoys	19
Crypto Hashes	20
Public Key Encryption	21
Private Key Encryption	22
Key Exchange Schemes	23
MySQL	24
PHP	25
Example Code	26
More Example Code	28
Yet More Example Code	30
Questions?	31